

# Configuring Redis Device Monitoring

The information in this article is for use with the Redis key-value store. Redis is a NoSQL document oriented database that stores data in the system's memory to leverage RAM speed, and is often used in situations that require high-availability and scalability. For more information on Redis, visit the software's website: <http://redis.io>

Ubersmith's device monitoring is extremely versatile, and by setting up a custom script for use with the TCP monitor, you can even issue commands to Redis to ensure it's running and meeting required conditions.

First, you'll want to head to Monitor Scripts in *Settings* so we can define the commands we'll send to Redis.

Click Add New Monitor Type, and give it a name to use as an identifier when we set up the monitor later on. In this example, we're sending the following commands:

## Commands:

*Send single line (with \n): AUTH password*

*Match Regular Expression: ^\|+OK\$*

These commands send and confirm that the authentication is confirmed properly (if necessary). The password can be customized on a per-monitor basis, since the script is loaded

*Send single line (with \n): PING*

*Match Regular Expression: ^\|+PONG\$*

These commands confirm that the Redis server is up and running on the host.

*Send single line (with \n): DBSIZE*

*Match Regular Expression: ^\|:([0-9]|[1-9][0-9]|[1-9][0-9][0-9])\$*

These commands will ensure that the database contains a certain amount of keys, and can be used to confirm that a background process is moving keys from Redis to less volatile storage on disk by ensuring the stored keys stay below a certain number. In this example, we are using regex to confirm that the number of keys stored is less than 1000.

*End of Script*

Save this script and name it accordingly, then head over to the Device Manager to set up the monitor on a device.

\*There are many other commands you can send to Redis other than the ones in use here. Visit their commands documentation for a full list of commands: <http://redis.io/commands>

Select a device, and click the Monitors tab, then use the Add Monitor link.

Select TCP for the protocol, and select or enter in the IP address that you're looking to monitor. Make sure you also specify the port Redis is listening on (default being 6379). Include your notify address and desired delays and intervals.

The config tab will automatically load the Redis script presets. This is where you can set the password if necessary, or remove commands that are not necessary for the particular host.

Once done with any script customization, click the save button, and your monitor will be set up. You'll probably want to test it to ensure it's reaching Redis correctly, and if so, the test will return success for each command. You can do so by using the test link on the monitor list.